Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2014) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

23rd International Meshing Roundtable (IMR23)

Block-Structured Hexahedral Meshes for CAD Models using 3D Frame Fields

N. Kowalski[a,*], F. Ledoux[b], P. Frey[c]

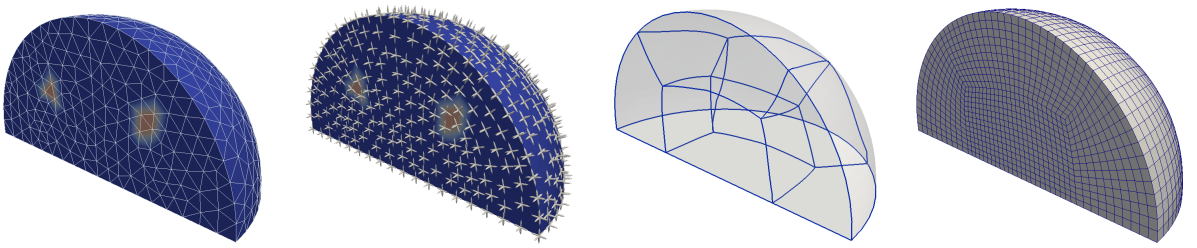
[a] *Université Catholique de Louvain, Avenue Georges Lematre, 4, B-1348 Louvain-la-Neuve, Belgium*
[b] *CEA, DAM, DIF, F-91297, Arpajon, France*
[c] *Université Pierre et Marie Curie, 4 place Jussieu, 75005, Paris, France*

Abstract

In the past couple of years, techniques using 3D frame fields have emerged to design hexahedral meshes[1,2]. Those methods are based on a two-step process where a 3D frame field is built by assigning a frame to each cell of a tetrahedral mesh, then a parametrization algorithm is applied to generate a hexahedral mesh. In this paper, we propose a novel algorithm to generate block-structured hexahedral meshes for any CAD domain Ω. This work differs from previous ones in several points: (1) the proposed approach does not start from a pre-meshed boundary; (2) The frame field initialization does not put singularity lines around the medial object of Ω; (3) Frames are assigned to the vertices and not to the cells of the tetrahedral mesh; (4) We do not perform a parametrization process but we generate a block structure that partition Ω in meshable regions.

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

Keywords: Mesh; Hexahedra; block-structured; Frame fields



(a) (b) (c) (d)

Fig. 1. Main steps of the proposed approach. In (a), the tetrahedral mesh through which the geometry is known. In (b), a 3D frame field is generated. In (c), a singularity graph is extracted leading to a block structure. In (d), a mesh obtained from the block structure.

*Corresponding author. Tel.: +33-664-321-729.
E-mail address: kowalski.nico@gmail.com

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

1. Introduction

Depending on the numerical approximation methods, hexahedral meshes are preferred to tetrahedral meshes due to their layered structure that can be aligned along the boundary of 3D domains. Moreover, they have interesting numerical properties, such as a reduced number of elements and a high approximation accuracy in numerous simulations in physics and mechanical engineering. However, generating high-quality hexahedral meshes is a very difficult and time-consuming task. Indeed, meshes are expected to abide by a number of rules to be of any use [3]: (1) To align elements along the boundary and inner directional constraints. This means having hexahedra layers along each boundary or constrained surface, and rows of hexahedra along each boundary or constrained edges. (2) To maximize the hexahedra quality by keeping their distortion to a minimum. To achieve this, singular edges, i.e. edges with more or less than four incident hexes, have to be introduced inside the volume to reduce global distortion, while being kept at a minimum to reduce local distortion. The work presented in this paper is motivated by generating such meshes.

We consider that the automatic generation of high-quality hexahedral meshes requires to have a geometric information inside the volume and not only on the boundary. Such an information can be provided by a frame field, as shown in [1,2,4]. In [4], authors provide theoretical foundations to characterize frame fields that are suitable for hexahedral mesh generation but their frame field computation is driven by a meta-mesh that must be provided by the user. In [1,2], the frame field is generated by solving a highly non non-linear energy function. A caveat of such methods is the impact of the initial solution that leads to a solution corresponding to local minima of the objective function. In both works, the initial solution is obtained through a simple straightforward process that can lead to non-optimal final solution:"...*propagation-based frame field initialization likely generates singular edges around the medial axis of the volume, and most of them cannot be eliminated by frame optimization*"[2]. This leads to potential high distortions for the resulting hexahedral elements, or even to unnecessary singularities.

In this paper, we present a method to generate block structures that can be used to generate full-hexahedral meshes. Starting from a tetrahedral mesh T_Ω of a domain Ω with sharp features, we build a continuous frame field F_Ω on T_Ω as a continuous piecewise linear frame field extending a discrete unit frame field that we compute on the vertices of T_Ω. We extend the approaches proposed in [1,2] by providing a novel initialization step avoiding clusters of singularities in the vicinity of the medial object of Ω. Then a singularity-graph is extracted from F_Ω to partition Ω is easy-to-mesh parts. Our approach differs from those proposed in [1,2] on many points:

1. The quadrilateral mesh of the surface is not a parameter of our approach. We start from a geometrical domain Ω and a tetrahedral mesh T_Ω of this domain. Both the surface and the volume mesh of Ω are generated by our approach.
2. Internal 3D frames are not defined by taking the nearest 3D frame defined on the boundary, but by solving an iterative process, which consists in defining internal 3D frames that are in stable areas as first. This is essential to control the location of singularities, since the optimisation problem that we will consider is non linear and the result strongly depends on the initial solution.
3. 3D frames are not associated to tetrahedral elements but to the vertices of T_Ω. As a consequence, we do not have a discrete singularity graph made of vertices and edges of T_Ω, but a singularity graph, which is built by applying linear interpolation into each tetrahedron containing a singularity of the 3D frame field. By this way, we avoid the topological cleaning process.
4. We do not generate an atlas of parametrizations, which is very sensitive and expensive to compute. We use the singularity graph to define a domain partitioning and then a block structure, where each bloc can be easily meshed using a simple mapping algorithm.

.

1.1. Related Work

All-hexahedral mesh generation has been widely studied for decades. A comprehensive survey is available in [5]. However, automatic algorithms providing good quality meshes for any domain are yet to be designed. As a result, semi-automatic methods based on robust algorithms designed for restrictive categories of domains, like multi-sweeping[6,7], were developed to help decomposing domains into meshable regions. Thus, the automatic generation

of hexahedral meshes for any geometric 3D object is still an open problem. Starting from a pre-meshed boundary surface, pure geometric approaches like plastering [8] or H-morph [9] algorithms have been proposed, while some other works have been based on pure topological approaches [10,11]. In both cases, the success was limited: some unfilled cavities remain or inverted cells and negative Jacobian cells can be generated. By relaxing the constraint of starting from a pre-meshed boundary that must be preserved, grid-based algorithms [12,13] or unconstrained plastering [14] provide encouraging examples of all-hexahedral meshes for arbitrary domains. Currently, grid-based [12,13] methods offer the simplest and most robust approach, but as the PolyCube method[15,16], it suffers from two flaws : the influence of the box orientation on the final result, leading to unpredictable results, and the location of the worst quality elements near the boundary while the quality of elements in this area is often critical for numerical approximation methods.

Quad mesh generation methods have recently been improved by using surface parametrization techniques on cross fields that are defined on a triangular surface [17,18]. Another approach consists in using the cross field to partition the domain into quadrilateral shaped blocks[19]. While the attempts to extend the former to hexahedral meshing have shown promising results [2,4], a generalization of the latter to 3D has yet to be designed. The Morse-Smale complex technique[20] provides another approach to quad meshing that has yet to be extended in a robust all-hexahedral meshing algorithm. The interested reader can see [21] for a recent survey.

Frame field has several definitions in the literature; for the purpose of this paper, we consider a frame to be the 3D extension of a cross, that is 3 directions orthogonal one to the other. Cross fields have been widely studied with applications to surface meshing and texture mapping. 3D Frame fields, on the other hand, have only been studied for the last few years. They are a guideline to drive meshing algorithms by providing geometric orientations inside the volume and not only on the boundary. Several papers [1,2,4,22] provide very interesting results. Nieser *et al.*[4] generate a frame field based on a given meta-mesh. Huang *et al.* [1,22,23] and Li *et al.* [2] provide a way to smooth a given frame field by minimizing an energy function, but their methods use rough initializations that can lead to issue that cannot be removed with a smoothing step.

1.2. Approach and Paper Outline

Starting from a tetrahedral mesh T_Ω that discretizes a geometric domain Ω, our aim is to generate a smooth discrete frame field where a single frame is associated to each vertex of T_Ω. Definitions about frame fields are given in Section 2. Our approach is decomposed in two main steps that will be respectively described in Sections 3 and 4 (see Fig 1): (1) First a 3D frame field is generated. In order to assign a frame to every vertex of T_Ω, we follow an advancing-front algorithm that consists in introducing frames in such a way that the smoothest areas of Ω will be filled at first. All the operations performed during this step are based on a quaternion representation of the frame that allow us to perform frame interpolation. A final global smoothing is performed as in [2]. This process relies on the representation of frames by Euler angles. (2) Then a singularity-graph defining the 3D frame field topology is extracted. As the considered frame field is obtained by linearly extrapolating a per-vertex discrete unit frame field, singularity lines are not a set of edges of T_Ω but are smoothly defined into the tetrahedral elements of T_Ω. Boundary singularity lines are computed in a second time. Before concluding, Section 5 provides some experimental results we obtained.

2. 3D Frame fields and hexahedral meshing

2.1. 3D frames to model hexahedra orientations

We define a 3D frame $\mathbf{F}$ as being a 3-uple $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$, with $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ three unit 3D vectors such that $\mathbf{u}.\mathbf{v} = 0$ and $\mathbf{w} = \mathbf{u} \wedge \mathbf{v}$. Considering a hexahedral element H, which has 3 main directions linking its pairwise opposite faces, 24 frames can thus represent it. This set of frames is invariant under rotations of $\frac{\pi}{2}$ around one of its three axis and forms an equivalence class: let $\mathbf{F}$ be a frame, we note this class $[\mathbf{F}]$. In other words, it corresponds to the cubical symmetry group $\mathcal{G}$ (any map in SO(3) which maps coordinate axes to coordinate axes). Such a definition of 3D frames is an intuitive 3D extension of crosses or 4-rosys as defined in [18] and that are invariant under $\frac{\pi}{2}$ rotations. But, while a 4-rosy can be represented by an unique 2D vector that allows to perform well-founded mathematical computation[24], no such representation vectors can be exhibited in 3D. In order to perform operations on frames, we use a group law

+ that combines any two elements $\mathbf{F}_a$ and $\mathbf{F}_b$ of an equivalence class $[\mathbf{F}]$ to form another element of $[\mathbf{F}]$. For instance, let $\mathbf{F}_i$ and $\mathbf{F}_j$ be two frames of $[\mathbf{F}]$, the linear combination $\alpha\mathbf{F}_i + \beta\mathbf{F}_j$ must be a component of $[\mathbf{F}]$ (see Figure 2). To define this law, we use unit quaternions to represent frames.

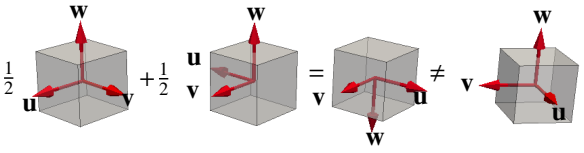


Fig. 2. Adding two equivalent frames must provide a third equivalent frames and not be computed as the average of the frames vectors.

Considering $(1, \mathbf{i}, \mathbf{j}, \mathbf{k})$ as being a basis of $\mathbb{R}^4$ such that $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$, a unit quaternion $\mathbf{q}$ is defined by $\mathbf{q} = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ with $w^2 + x^2 + y^2 + z^2 = 1$. For any frame $\mathbf{F}_i = \{\mathbf{u}_i, \mathbf{v}_i, \mathbf{w}_i\}$ is associated the unit quaternion $\mathbf{q}_i$ that corresponds to the rotation matrix transforming $\{\mathbf{x}(1,0,0), \mathbf{y}(0,1,0), \mathbf{z}(0,0,1)\}$ into $\{\mathbf{u}_i, \mathbf{v}_i, \mathbf{w}_i\}$. Let $\mathbf{F}_i$ to $\mathbf{F}_j$ be two frames we want to compare. Let $\mathbf{q}_i$ and $\mathbf{q}_j$ be their associated quaternions, then $\mathbf{q}_i{\mathbf{q}_j}^{-1}$ corresponds to transforming $\mathbf{F}_j$ into $\mathbf{F}_i$. Then we define the distance d_{ij} between two quaternions $\mathbf{F}_j$ to $\mathbf{F}_i$ as

$$d_{ij} = \min_{\mathbf{g}\in\mathcal{G}} 1 - \mid \mathbf{g}.(\mathbf{q}_i{\mathbf{q}_j}^{-1}) \mid \tag{1}$$

where $\mathbf{g}$ is a quaternion corresponding to a rotation of $\mathcal{G}$. Then $\mathbf{F}_j \in [\mathbf{F}_i]$ if and only if $d_{ij} = 0$. Note that we do not compute an accurate geometric distance on the sphere in $\mathbb{R}^4$ but a simple dot product which is less expensive in terms of computational cost.

We also use quaternions to perform linear interpolation between frames. Let $\mathbf{F}_i$ and $\mathbf{F}_j$ be two frames, the frame $\mathbf{F}_{ij} = \alpha\mathbf{F}_i + \beta\mathbf{F}_j$, with $\alpha > 0, \beta > 0$ and $\alpha + \beta = 1$, the quaternion associated to $\mathbf{F}_{ij}$ is

$$\frac{\alpha\mathbf{q}_i + \beta\mathbf{g}\mathbf{q}_j}{|\alpha\mathbf{q}_i + \beta\mathbf{g}\mathbf{q}_j|}$$

with $\mathbf{g}_m$ satisfying Equation 1, and $\mathbf{g} = \mathbf{g}_m$ if $\mathbf{g}_m.(\mathbf{q}_i{\mathbf{q}_j}^{-1}) > 0$, $-\mathbf{g}_m$ otherwise. This formulation can be extented to any number of frames.

We also represent frames by Euler angles to perform a global smoothing of the frame field (as in [2]). The main benefit of this representation is to intrinsically constraint solutions onto the unit sphere. On the contrary, solving systems using the 4 coordinates of the quaternions would imply using extra constraints to preserve their unit norm. In fact, those representations are complementary. Let a 3D frame $\mathbf{F}$ and $\mathbf{q}$ its associated unit quaternion. Every unit quaternions represents a rotation that can be decomposed using Euler angles. We use an XYZ decomposition where $\mathbf{q} = \mathbf{q}_\mathbf{x}^\alpha\mathbf{q}_\mathbf{y}^\theta\mathbf{q}_\mathbf{z}^\gamma$, with α the rotation angle around axis $\mathbf{x}$, θ the rotation angle around axis $\mathbf{y}$ and γ the rotation angle around axis $\mathbf{z}$.

2.2. Discrete and continuous frame fields

Given a tetrahedral mesh T_Ω, we define a *discrete frame field* by assigning a frame to each vertex of T_Ω. This field can be extended to a *continuous frame field* on T_Ω as a piecewise linear interpolation in each tetrahedron of T_Ω. In such a field, the magnitude of frames can be non-unitary and singularities are located in points of T_Ω where frame magnitude is zero. Extending a discrete unit frame field to a continuous non-unit one ensures to this latter to have singularities that are gathered in a network of lines, similar to the *singularity graph* in [4].

Proposition 1. *Considering a continuous frame field F on T_Ω built from a unit frame field defined on the vertices of T_Ω, a singularity of F cannot be isolated.*

Proof: Due to the linear interpolation process used to build frames inside tetrahedra, having a singularity inside a tetrahedron T means that one of the triangles adjacent to T contains a singularity too. The reciprocate is true. Therefore, singularities can not be isolated and are structured in lines of singularities. For classical vector fields,

a singularity is located where the magnitude of a vector is zero. A singularity is then characterized by an integer number, its index, describing the behavior of the field in the vicinity. On 2D manifolds, this notion was generalized to frame-fields [17,25] and used to show that the valence of a singularity is either equal to 3 or 5 in 4-rosy fields[19].

Proposition 2. *Considering a continuous frame field F on T_Ω built from a unit frame field defined on the vertices of T_Ω, the degree of singularity lines is 3 or 5.*

The proof of this proposition is similar to the proof provided for the 2D case in [19] by considering each face of each tetrahedron individually. As we assign frames to vertices and not to tetrahedral elements, a singularity line either ends on $\partial\Omega$, inside Ω or meets other singularity lines inside the volume Ω. Having singularity lines that end inside Ω can not happen with the approaches of [1,2,4]. Moreover, we assign frames to the vertices of T_Ω to generate a continuous frame field over Ω by piecewise linear interpolation, while tet-assigned frames are used in [1,2,4] to define a discrete singularity graph composed of *singular* edges of T_Ω. As a consequence, we do not have to perform topological cleaning to get smoother singularity lines.

3. Boundary-aligned frame field generation

The first part of the proposed method consists in generating a frame field starting from a tetrahedral mesh T_Ω without any quadrilateral mesh of $\partial\Omega$. Initially, frames are fully defined onto the geometric curves of Ω, i.e lines of boundary edges where the dihedral angle between adjacent boundary triangle is greater than a threshold and frames are partially defined onto the remainder of $\partial\Omega$. The left part of Figure 3 illustrates it. In (a), frame $\mathbf{F}$ is initially aligned with the tangent to the curve in V and the average of the normal to the triangles in the vicinity of V. In (b), only the red component of frames $\mathbf{F}_1$ to $\mathbf{F}_3$ is initially aligned with the normal to $\partial\Omega$. Other components of these frames will be defined during the frame field generation.

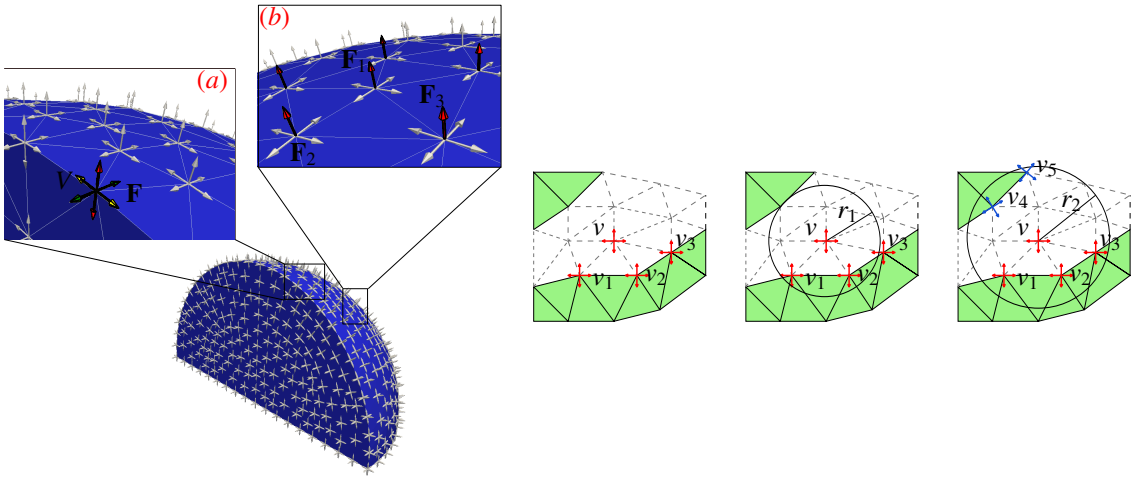


Fig. 3. On the left, an example of frame field with two focus on initial conditions: in (a), a frame $\mathbf{F}$ is completely defined at the beginning, while only one component of the frames depicted in (b) is known. On the right, balls are depicted to show the frames involved to evaluate the smoothness in the vicinity of v. As balls grow, the smoothness decreases.

In order to obtain a smooth field, we perform a global smoothing similar to the one done in[2], which consists in minimizing an energy function. This function is highly non-linear and thus the result strongly depends on the initial values that are assigned to each vertex of T_Ω. In the first part of this section, we present an initialization process, which yields empirically lower energy than previous initialization methods [1,2,4]. In the second part, we describe how the global smoothing proceeds.

3.1. Selection of the frame to insert

In order to initialize the unit discrete frame field, we use an advancing-front algorithm that can be sum up as follows:

1. Starting with a small set of vertices S_V of T_Ω for which frames are already assigned (on the geometric curves of Ω at the very beginning), select a new vertex v adjacent to a vertex of S_V;
2. Interpolate the frames of the vertices of S_V adjacent to v in order to define the frame at v;
3. Repeat steps 1 and 2 until a frame is assigned to every vertex of T_Ω.

The order in which vertices are selected vastly impacts the location of singularities. Indeed, due to the interpolation process, singularities will tend to be close to the last inserted vertices. Moreover, as the energy computed in the smoothing part of the algorithm is non-convex, the reached minima is local and not global. In practice, this means that the location of singularities will remain roughly the same after the smoothing step. In our case, we select the vertex v that is located in the steadiest area of Ω in the meaning of a measure of the field smoothness. To define this smoothness measure let us introduce some notions. The right part of Figure 3 illustrates these notions in 2D. Let v be a candidate vertex to add to S_V, i.e a vertex adjacent to at least one vertex of S_V. We note $\mathbf{F}_v$ the local frame computed in v by interpolating the frames assigned to the vertices of S_V adjacent to v. The weight associated to each frame $\mathbf{F}_{v'}$ that participates to define $\mathbf{F}'_v$ is function of the distance between v' and v. We denote $B_{v,r}$ the ball of center v and radius r and we define the function $d(B_{v,r})$ as

$$\max_{v_i \in B(v,r)} d(\mathbf{F}_i, \mathbf{F}_v) \tag{2}$$

where $\mathbf{F}_i$ is the frame assigned to the vertex v_i and the distance between frames $\mathbf{F}_i$ and $\mathbf{F}_v$ is computed using Equation 1. To select the vertex v to insert into S_V, we can not arbitrarily choose the value of r. Thus, considering R as being the radius of the bounding box of Ω, we compute for each candidate vertex v of T_Ω the quantity

$$\int_{r=0}^{R} \max_{v_i \in B(v,r)} d(\mathbf{F}_i, \mathbf{F}_v) dr. \tag{3}$$

This value provides an estimation of the field smoothness at v : if the field is steady on a large area around v, then there exists a value r' such that Equation 2 is null for any $r < r'$. When a difference between frames happens at a distance r_i, it contributes accordingly to the amplitude of this change. This estimation provides good results when $\partial\Omega$ is mostly planar, but leads to the insertion of singularities in the close vicinity of spherical part of $\partial\Omega$. To alleviate this problem, the result of Equation 3 for a vertex v is weighted by the distance between v and $\partial\Omega$. This enforces the algorithm to first insert the points closer to $\partial\Omega$. Figure 4 illustrates how S_V is progressively built. It can be seen that some vertices located near the medial axis are inserted early in the process due to being in a zone where the field is steady.

Remark. The complexity time of this initialization step is at worst $O(n^2)$ with n the number of vertices in T_Ω.

3.2. Frame Field Optimization by global smoothing

The frame field is smoothed in a similar way as in [2]. Given two frames $\mathbf{F}_1$ and $\mathbf{F}_2$ and their respective quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$, the quaternion $\mathbf{q} = \mathbf{q}_2\mathbf{q}_1^{-1}$ transforms $\mathbf{F}_1$ into $\mathbf{F}_2$. Considering the corresponding rotation matrix $M_{1\to2}$, a measure of the closeness between $\mathbf{F}_1$ and $\mathbf{F}_2$ can be defined as :

$$E_{12} = \sum_i [H(M_{1\to2}[., i]) + H(M_{1\to2}[i, .])] \tag{4}$$

where $M_{1\to2}[., i]$ and $M_{1\to2}[i, .]$ denote the i^{th} row and column vector respectively, and $H(\mathbf{v}) = \mathbf{v}_x^2\mathbf{v}_y^2 + \mathbf{v}_y^2\mathbf{v}_z^2 + \mathbf{v}_z^2\mathbf{v}_x^2$. The function H is invariant on $[\mathbf{F}]$. This measure can be expressed on any edge e_{ij} of T_Ω between the frames of its end vertices v_i and v_j. We get the following energy function for the whole field :

$$E = \sum_{e_{ij} \in T_\Omega^E} E_{ij} \tag{5}$$

Fig. 4. A domain at different stages of the frame assignement process. All the vertices of red tetrahedra have an assigned frames, while blue tetrahedra have at least one vertex without any frame.

where T^E_Ω is the set of edges of T_Ω. However, trying to minimize this energy right away would generate general matrices and not rotational ones. Instead of applying a post process that would consist in projecting the obtained value onto the unit sphere, and thus introducing numerical errors, we use Euler angles, as done in [2]. Inner and boundary frames are then distinguished. An inner frame $\mathbf{F}_i$ has three degree of freedom: each of the matrices representing it can be written

$$M_i = R_{\mathbf{x}}(\alpha_i)R_{\mathbf{y}}(\theta_i)R_{\mathbf{z}}(\gamma_i)$$

where α_i, θ_i and γ_i are the rotation angle around axis $\mathbf{x}$, axis $\mathbf{y}$ and axis $\mathbf{z}$ respectively. A boundary frame at a vertex V has only one degree of freedom θ. It can be expressed

$$\{cos(\theta)\mathbf{T_1} + sin(\theta)\mathbf{T_2},\ -sin(\theta)\mathbf{T_1} + cos(\theta)\mathbf{T_2},\ \mathbf{N}\}$$

with $\mathbf{N}$ the normal to $\partial\Omega$ at V and $\mathbf{T_1}, \mathbf{T_2}$ two orthogonal tangent vectors relying in the tangent plane of $\partial\Omega$ at V. The minimization of the nonlinear function E of Equation 5 is performed using the same L-BFGS method than the one used in [2]. This final step improves the generated field but not in the same proportions than [1]. Indeed singularities are better located at the end of our initialization process. Consequently, E converges much quicker, in 10 to 25 iterations instead of several hundreds, confirming that the proposed initialization step leads to a better global solution in terms of the energy function of Equation 5.

3.3. Benefits of the initialization process

The rough initialization process proposed by [2] can lead to frame fields where singularity lines are along the medial object of Ω. In fact, as their approach starts from a quadrilateral mesh of $\partial\Omega$, the resulting singularity graph strongly depends of this boundary mesh. Left and middle of Figure 5 shows an extreme example of how singularities can be gathered around the medial object. In this case, even after a global smoothing, the obtained field has singularity lines that are too closely intertwined to be of any use. Our initialization process, on the other hand, generate singularity lines that are separated one from another (top row of Figure 5).

We also evaluated our initialization process in 2D. The right part of Figure 5 shows the resulting fields obtained either using our approach (top) or by assigning the vertices by only considering their distance to the boundary (bottom). In the second case, the two singularities are clustered on the medial object. While the global smoothing that follows the initialization step is able to improve the results somewhat, the resulting field is still very distorted, and the

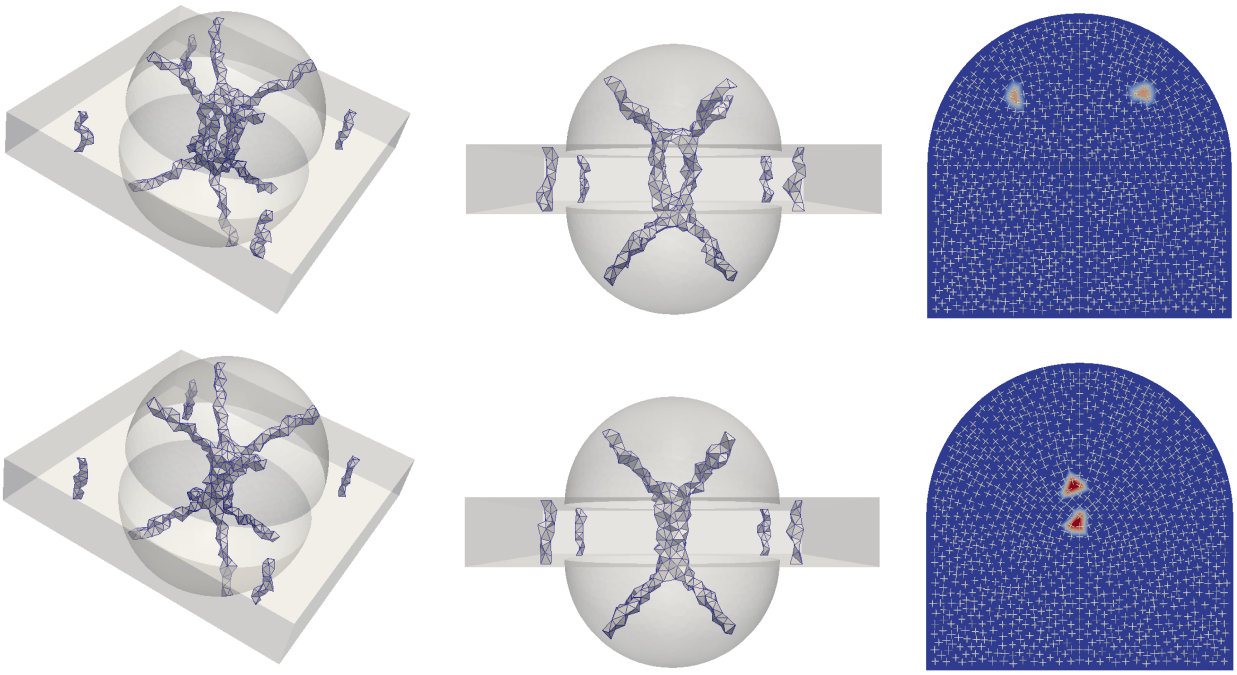


Fig. 5. Tetrahedral elements traversed by singularity lines of the 3D frame field with the proposed initialization process (top row, left and middle) and a simple initialization where the value of a frame in Ω is initialized with the value of the frame int the closest vertex on $\partial\Omega$ (bottom row, top and middle). For comparison, on the right, we show the 2D singularity placement with our initialization process (top) and an initialization process that assigns to any innner vertex the frame defined on the nearest boundary vertex (bottom)

singularities are still very close one to the other. It is due to the fact that the global smoothing could only find the closest local minima of the energy function. On the other hand, our approach gives a natural placement of singularities, and provide results consistent both with the usual manual decomposition of such a domain and the results obtained through state-of-the-art algorithms ([17–19]).

4. Singularity graph extraction

The second step consists in extracting the topological features of the frame field to build a singularity graph. Figure 6 illustrates the global process that we detail in the following. We first detect and build inner singularity points p_1 and p_2 and inner lines. Then, for each boundary singularity points, boundary singularity lines are built leading to an incomplete partitioning of $\partial\Omega$. Finally, boundary singularity lines are connected and inner surface patches can be derived.

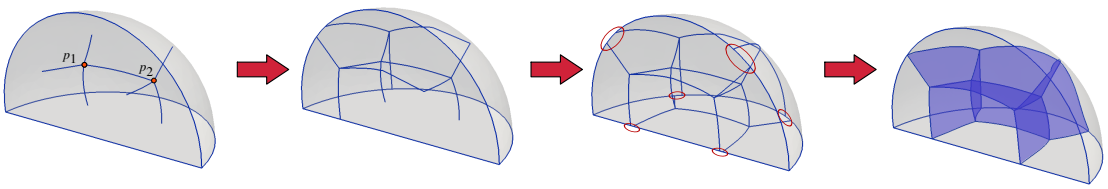


Fig. 6. *The singularity graph is built in an iterative way by propagating singularity lines inside Ω then on $\partial\Omega$ before partitioning Ω in several blocks.*

4.1. Detection of the inner singularity points and lines

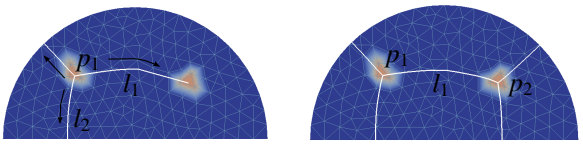
Defining a continuous frame field F on T_Ω starting from a discrete unit frame field defined on the vertices of T_Ω implies that we do not have isolated singularity point. As a consequence, we detect singularity points and lines in T_Ω by analyzing faces of T_Ω. To detect if a face f is traversed by a singularity line, we consider the behavior of the frame field F along the loop defined by the adjacent edges of f: frames defined at the vertices of f are projected onto the 2D plane of f to define a local cross field where a singularity index, similar to the notion of Poincaré index for vector fields, can be computed [17,25].

Considering a tetrahedral element $t \in T_\Omega$, either one, two or three of its faces can be traversed by a singularity line l_s. In the first case, l_s ends up into t meaning that we can not derive a block structure for Ω. In the second case, t is traversed by l_s and the intersecting points between l_s and t can be computed by finding the zeroes of the frame field in each traversed face of t. In the latter case, we have a singularity point, that is a meeting point of singularity lines in t or in a cluster of tetrahedra including t.

Singularity lines are then built as piecewise linear lines joining two inner singularity points or an inner singularity point to $\partial\Omega$. The list of points defining a singularity line l_s is made of the intersection points between the faces of T_Ω and l_s. We apply a post process smoothing algorithm to obtain smoother curves.

4.2. Extraction of the boundary singularity lines

Once inner singularity lines defined, boundary singularity lines must be built. Starting from each boundary singularity point p_i, i.e. a boundary ending point of an inner singularity line, we spread up singularity lines from p_i onto $\partial\Omega$. To do it, we use a fourth order Runge-Kutta method, locally to each crossed triangle $T \in \partial\Omega$. This process is used to spread all the boundary singularity lines emanating from a boundary singularity point. Such lines end up either on another boundary singularity point (like the line l_1 in Fig 7-*a*) or on a geometric curve (like the line l_2 in Fig 7-*a*). If it ends in a singularity point, the line is spread again in the opposite direction and blend with the previous one to ensure that the process does not depend on the order in which singularities are treated (see Fig 7-*b* where the line l_1 is different of the one presented in a). Technically, when we spread a boundary line from a boundary point, it can meet another singular point or miss it due to numerical approximations and the refinement level of T_Ω. We thus consider that a boundary line end up on a singular point whenever it comes close enough to it during the integration process. The distance at which we reconnect them should be determined according to the size of the mesh that is desired, as it can change the partitionning (see Fig 8). Getting a robust algorithm is then very tricky even if there is no theoretical limitation.



(*a*) (*b*)

Fig. 7. Boundary singularity lines built on a surface. Lines starting in p_1 are given in (a), while lines coming from p_2 are given in (b). Note that the linking line between p_1 and p_2 was built twice and then blended to obtain a symmetric process.

4.3. Propagation and connection of the boundary singularity lines

First boundary singularity lines are spread up from boundary singularity points. Then a boundary singularity line l_s connects boundary singularity points or reach a geometric curve C defined as a line of edges on $\partial\Omega$. In this case, l_s is propagated on the other side of C or connects to a line l_s^2 defined on the other side of C. In Figure 9-a, the boundary lines defined on the two sides of C are not connected. Then left lines are connected in b and right ones in c. Note that the criterion to connect lines to singularity points or boundary points is once again based on a threshold value that may change the topology of the partitionning (see Fig 8).

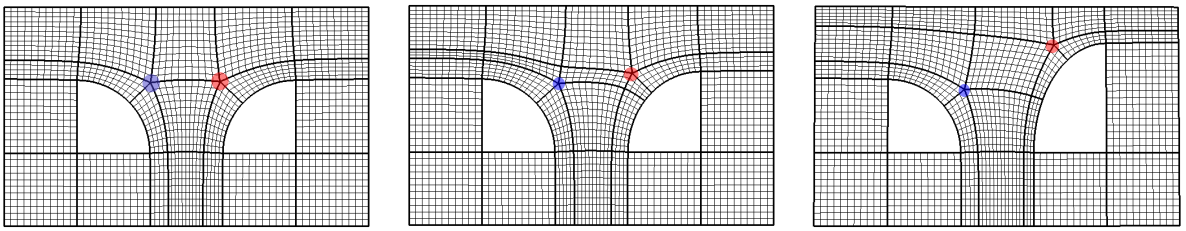


Fig. 8. Depending of some slight variations in the geometric domain boundary, singularity lines do not necessarily connect the red and blue singularity points. In (a), they are connected, while in (b) and (c) they are not.

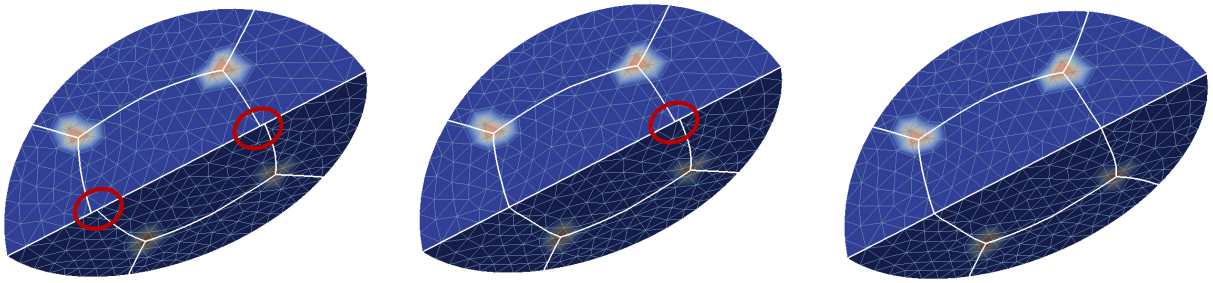


Fig. 9. Connection of boundary singularity lines that reach a common curve

5. Experimental results

Our approach has been evaluated on various tetrahedral meshes that discretize CAD models. There is no boundary frame field provided as an input. Figure 10 and Figure 11 illustrate the impact of the mesh resolution onto the algorithm. From left to right on Figure 10, meshes have respectively 5738, 58871 and 175079 cells. We can see that the tetrahedra set containing singularity lines becomes sharper and thinner demonstrating that our method seems to converge when the geometric model is better approximated. On Figure 11, the obtained singularity graphs are similar: the algorithm is thus robust enough as soon as the domain is sufficiently approximated.

Figures 13, 14 and 12 show all-hexahedra meshes obtained for different domains. The singularity graph always defined a block structure adapted to hexahedral meshing. We can notice that the symmetry of the domains is not totally preserved in the generated frame fields. Better results could be obtained with more refined meshes.

6. Conclusions and Future Work

In this paper we presented a novel algorithm to generate block-structures that are adapted to hexahedral mesh generation. Following [1,2], a frame field is used to guide the block-structure creation but our work is original in many aspects: (1) the proposed algorithm does not require a pre-meshed boundary nor a predefined boundary cross field; (2) Using an optimized initialization step, we alleviate a drawback of [1,2], preventing lines of singularities to be clustered around the medial object and lowering the energy value of the field; (3) We use a per-vertex formulation that allows us to generate smooth singularity graphs without performing post-process topological operations on T_Ω; (4) Instead of performing a parametrization algorithm, a block structure is built on the singularity graph of the frame field.

Improvements and future works have to be done to get a more robust algorithm. For instance, it would be interesting to use an adaptive process where a coarse mesh would be used to get a first estimation of the solution before refining it in the vicinity of the singularity points and lines. Such an improvement could significantly reduce the computational cost of the initialization step. Moreover, the main drawback of the proposed approach is that we have no theoretical guarantee that the generated frame field corresponds to the structure of a hexahedral mesh. To achieve this, we believe

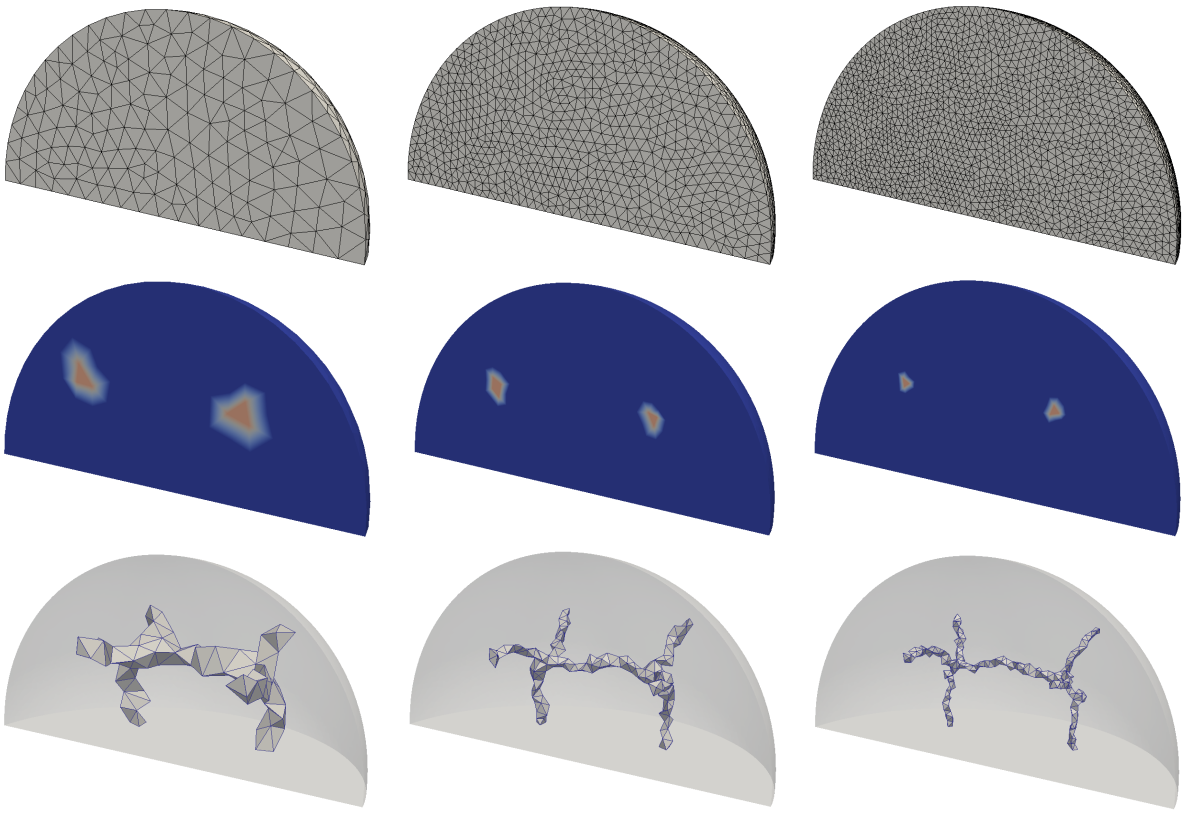


Fig. 10. Three approximations of a domain. Initial meshes are given in the first row; frame fiels obtained after the first step of the algorithms are given in the second row; In the third row, only the cells intersected by inner singularity lines are visible.

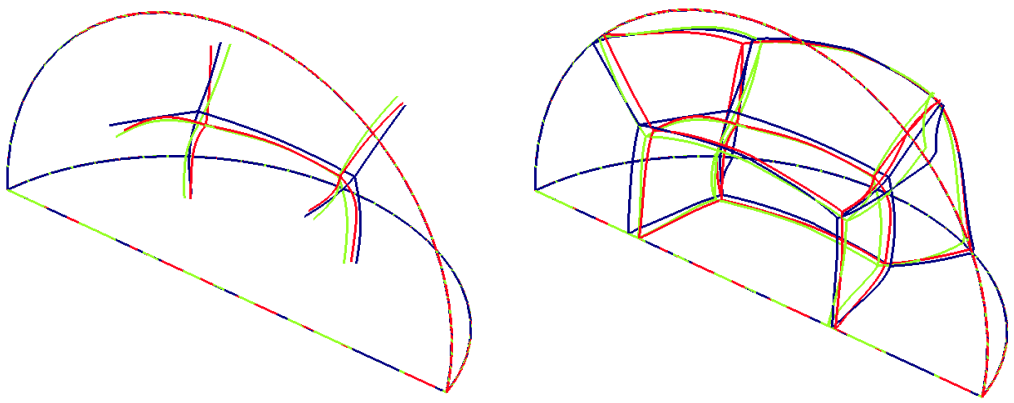


Fig. 11. Inner singularity lines and complete singularity graphs for the three meshes used in Figure 10.

that it is necessary to take into account the global structure of hexahedral meshes in the formulation of the energy to minimize.

Acknowledgments.. The authors would like to thank Y. Liu for kindly answering to their questions about [2].

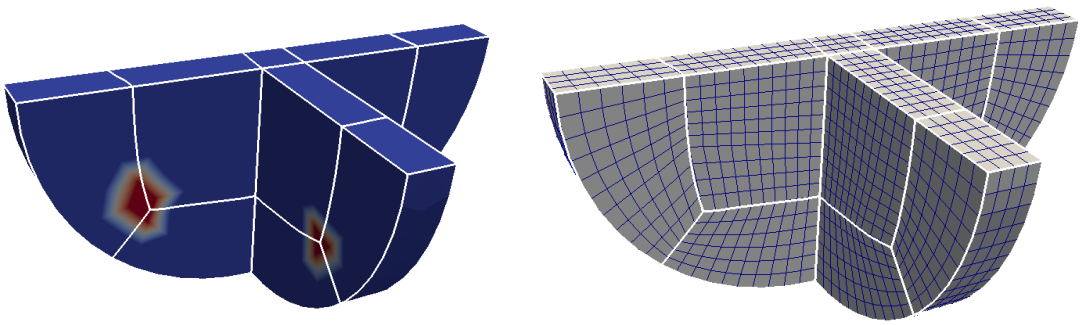


Fig. 12. Frame field in (a) and corresponding hexahedral mesh in (b) for a non-sweepable domain.

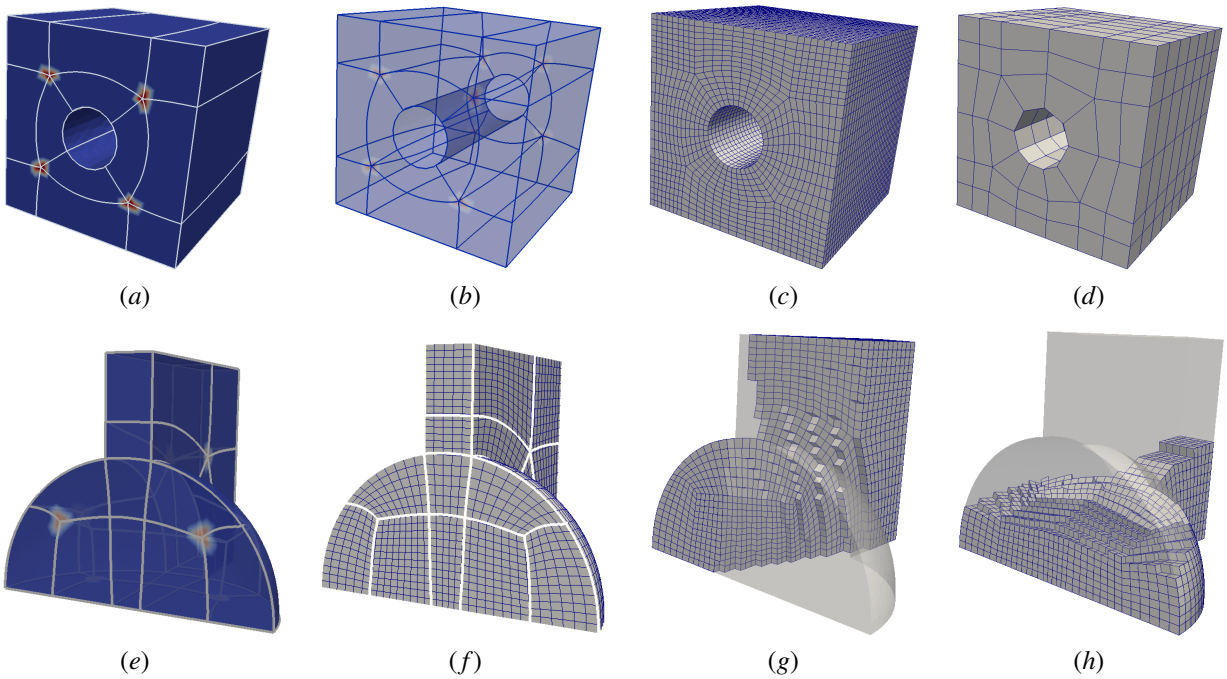
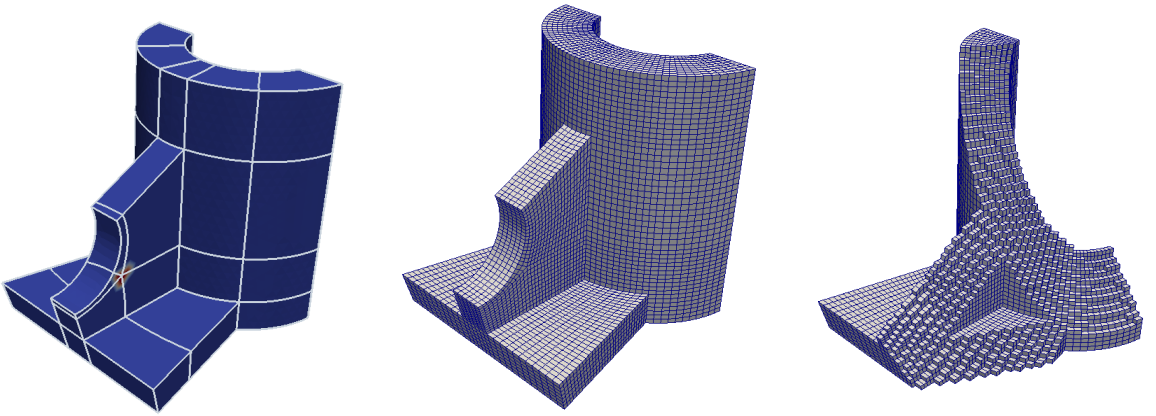


Fig. 13. In the first row, a symmetric domain where singularity points are not totally aligned between the front and the back face leading to a slightly twisted structure. In (a), the non-symmetry of the field on the front face can be seen and in (b) in the whole inner domain. In (c) and (d), two hexahedral meshes obtained from the block-structure with different levels of refinement are provided. In the second row, the singularity graph shown in (e) allows to generate a full hexahedral mesh that is presented in (f) with two innner cuts in (g) and (h).

References

[1] J. Huang, Y. Tong, H. Wei, H. Bao, Boundary aligned smooth 3d cross-frame field, ACM Trans. Graph. 30 (2011) 143:1–143:8.
[2] Y. Li, Y. Liu, W. Xu, W. Wang, B. Guo, All-hex meshing using singularity-restricted field, ACM Trans. Graph. 31 (2012) 177:1–177:11.
[3] F. Ledoux, J. Shepherd, Topological and geometrical properties of hexahedral meshes, Engineering with Computers 26 (2010) 419–432.
[4] M. Nieser, U. Reitebuch, K. Polthier, Cubecover- parameterization of 3d volumes, Comput. Graph. Forum 30 (2011) 1397–1406.
[5] T. J. Tautges, The generation of hexahedral meshes for assembly geometry: survey and progress, International Journal for Numerical Methods in Engineering 50 (2001) 2617–2642.
[6] P. K. J. Shepherd, S.A. Mitchell, D. White, Methods for multisweep automation, in: proceedings of the 9th International Meshing Roundtable, 2000, pp. 77–87.
[7] X. R. Eloi Ruiz-Gironés, J. Sarrate, A new procedure to compute imprints in multi-sweeping algorithms, in: B. W. Clark (Ed.), proceedings of the 18th International Meshing Roundtable, Springer, 2009, pp. 281–299.
[8] T. Blacker, R. Meyers, Seams and wedges in plastering:a 3d hexahedral mesh generation algorithm, Engineering with Computers 2 (1993) 83–93.



(a) (b) (c)

Fig. 14. The singularity graph shown in (a) corresponds to the hexahedral mesh shown in (b) and (c).

[9] S. Owen, S. Sunil, H-morph: An indirect approach to advancing front hex meshing, International Journal for Numerical Methods in Engineering 1 (2000) 289–312.

[10] T. Tautges, T. Blacker, S. Mitchell, The whisker weaving algorithm: A connectivity-based method for constructing all-hexahedral finite element meshes, International Journal For Numerical Methods in Engineering 39 (1996) 3327–3349.

[11] F. Ledoux, J.-C. Weill, An extension of the reliable whisker weaving algorithm, in: proceedings of the 16th International Meshing Roundtable, Springer, 2007, pp. 215–232.

[12] R. Schneiders, An algorithm for the generation of hexahedral element meshes based on a octree technique, in: proceedings of the 6th International Meshing Roundtable, 1997, pp. 183–194.

[13] L. Maréchal, Advances in octree-based all-hexahedral mesh generation: Handling sharp features, in: B. W. Clark (Ed.), proceedings of the 18th International Meshing Roundtable, Springer, 2009, pp. 65–84.

[14] M. L. Staten, R. A. Kerr, S. J. Owen, T. D. Blacker, M. Stupazzini, K. Shimada, Unconstrained plastering - hexahedral mesh generation via advancing-front geometry decomposition, International Journal for Numerical Methods in Engineering 81 (2010) 135–171.

[15] J. Gregson, A. Sheffer, E. Zhang, All-hex mesh generation via volumetric polycube deformation, Comput. Graph. Forum 30 (2011) 1407–1416.

[16] J. Huang, T. Jiang, Z. Shi, Y. Tong, H. Bao, M. Desbrun, L1-based construction of polycube maps from complex shapes, in: ACM Transactions on Graphics, 2014.

[17] F. Kalberer, M. Nieser, K. Polthier, Quadcover - surface parameterization using branched coverings, Comput. Graph. Forum 26 (2007) 375–384.

[18] N. Ray, B. Vallet, W.-C. Li, B. Lévy, N-symmetry direction field design, ACM Trans. Graph. 27 (2008).

[19] N. Kowalski, F. Ledoux, P. Frey, A pde based approach to multidomain partitioning and quadrilateral meshing, in: Springer-Verlag (Ed.), 21th International Meshing Roundtable, 2012, pp. 137–154.

[20] J. Huang, M. Zhang, J. Ma, X. Liu, L. Kobbelt, H. Bao, Spectral quadrangulation with orientation and alignment control, ACM Trans. Graph. 27 (2008) 147.

[21] D. Bommes, B. Levy, N. Pietroni, E. Puppo, C. Silva, M. Tarini, D. Zorin, Quad-mesh generation and processing: A survey, Computer Graphics Forum (2013).

[22] J. Huang, T. Jiang, Y. Wang, Y. Tong, H. Bao, Automatic Frame Field Guided Hexahedral Mesh Generation, Technical Report, State Key Lab of CAD&CG, College of Computer Science at Zhejiang University, 2012.

[23] T. Jiang, J. Huang, Y. Wang, Y. Tong, Frame field singularity correction for automatic hexahedralization, in: IEEE Transactions on Visualization and Computer Graphics, 2014.

[24] J. Palacios, E. Zhang, Rotational symmetry field design on surfaces, ACM Trans. Graph. 26 (2007) 55.

[25] M. Nieser, K. Polthier, Parameterizing singularities of positive integral index, in: IMA Conference on the Mathematics of Surfaces, 2009, pp. 265–277.